J. Symbolic Computation (1985) **1**, 7–29

Equational Methods in First Order Predicate Calculus

ETIENNE PAUL

Centre National d'Etudes des Télécommunications,
38/40 *Rue du Général Leclerc,*
92131 *Issy les Moulineaux, France*

(*Received* 12 *July* 1984)

We show that the application of the resolution principle to a set of clauses can be regarded as the construction of a term rewriting system confluent on valid formulas. This result allows the extension of usual properties and methods of equational theories (such as Birkhoff's theorem and the Knuth and Bendix completion algorithm) to quantifier-free first order theories. These results are extended to first order predicate calculus in an equational theory, as studied by Plotkin (1972), Slagle (1974) and Lankford (1975). This paper is a continuation of the work of Hsiang & Dershowitz (1983), who have already shown that rewrite methods can be used in first order predicate calculus. The main difference is the following: Hsiang uses rewrite methods only as a refutational proof technique, the initial set of formulas being unsatisfiable iff the equation TRUE = FALSE is generated by the completion algorithm. We generalise these methods to satisfiable theories; in particular, we show that the concept of confluent rewriting system, which is the main tool for studying equational theories, can be extended to any quantifier-free first order theory. Furthermore, we show that rewrite methods can be used even if formulas are kept in clausal form.

1. Introduction

We show in this paper that the resolution algorithm applied to a set of clauses has the same goal as the Knuth & Bendix algorithm applied to a set of equations, namely the construction of a confluent rewriting system, but with the restriction that the system obtained is confluent only on valid formulas.

More specifically, let S be a quantifier-free theory defined by a set of clauses. We show that the application of the resolution principle to S produces a rewriting system R such that any quantifier-free first order formula F which is a valid consequence of S reduces to TRUE, using reductions from R in any order. Conversely, only valid consequences of S reduce to TRUE. But the system R is not confluent in general, for F may possess several normal forms if it is not a valid consequence of S.

Such partly confluent rewriting systems have already been studied in equational theories: for example, the Greendlinger–Bucken algorithm (Bucken, 1979) for finitely presented groups constructs a rewriting system which, under certain conditions, is confluent on the relators (i.e. words equal to the identity in the group). The word problem can then be decided as follows: two words a and b are equal iff the normal form of the word $a.b^{-1}$ is the identity.

A preliminary version of this paper was presented as "A new interpretation of the resolution principle" at the 7th International Conference on Automated Deduction in Napa, California, May 1984. The proceedings are in Springer-Verlag, *Lecture Notes in Computer Sciences 170*, edited by R. E. Shostak.

0747-7171/85/010007+23 $03.00/0

In the same way, the equivalence between two quantifier-free formulas F and G in the theory S can be decided by computing the normal form of the compound formula $F \langle = \rangle G$: F and G are equivalent iff this normal form is TRUE.

From this first result, we obtain that usual properties and methods of equational theories (such as Birkhoff's theorem and the Knuth & Bendix completion algorithm) can be extended to any quantifier-free first order theory.

The organisation of the paper is as follows:

In Section 2, we introduce a rewriting system in propositional calculus confluent on valid formulas. Since we do not need general confluence, we will not use the system discovered by Hsiang (1983), because this system is not practical for manipulating clauses. Our system will be constructed from the usual Boolean connectors "and", "or", "not".

In Section 3, we extend this result to first order predicate calculus: we describe a completion algorithm based on the resolution principle which generates, from any initial set of clauses, a rewriting system confluent on valid formulas.

In Section 4, we prove Birkhoff's theorem for first order predicate calculus. This theorem states that equational-like deduction (i.e. deduction by instantiation and replacement of equivalents by equivalents), together with a suitable equational axiomatisation of propositional calculus, is complete for quantifier-free first order theories. From this theorem, we obtain another completion algorithm, based on the Knuth & Bendix algorithm and the Hsiang system for propositional calculus.

In Section 5, the previous results are extended to first order predicate calculus in an equational theory.

2. Confluence on Valid Formulas in Propositional Calculus

2.1. REVIEW OF EQUATIONAL TERM REWRITING SYSTEMS

It is assumed that the reader is familiar with the literature on equational theories and term rewriting systems. See Huet & Oppen (1980) for a full description.

We start with a vocabulary of variables and function symbols; we define terms, occurrences, substitutions, unification, most general unifier (mgu). If M is a term and u an occurrence of M, M/u denotes the subterm of M at occurrence u, and $M[u <- N]$ the term M in which this subterm is replaced by N. If σ is a substitution, the application of σ to M is denoted $M\sigma$.

An equational system E is a set of pairs $M = N$, where M and N are two terms. The equality relation generated by E, according to the usual rules of equational reasoning (Huet & Oppen, 1980), is denoted $=_E$.

A term rewriting system R is a set of directed pairs $l \to r$ such that each variable in r occurs in l. A term $t1$ R-reduces at occurrence u to a term $t2$ using the rule $l \to r$ iff there exists a substitution σ such that $t1/u = l\sigma$ and $t2 = t1[u <- r\sigma]$. We write $t1 \to_R t2$.

Given E and R as above, a term $t1$ E,R-reduces to $t2$ iff there exists a term $t3$ such that $t1 =_E t3$ and $t3 \to_R t2$. We write $t1 \to_{E,R} t2$. The relation $\to_{E,R}$ can also be regarded as the relation induced by R in E-equivalence classes of terms. $\overset{*}{\to}_{E,R}$ denotes the reflexive–transitive closure of $\to_{E,R}$, and $=_{E,R}$ denotes the reflexive–symmetric–transitive closure of $\to_{E,R}$.

The pair (E,R) is called an equational term rewriting system (ETRS). Such systems are a generalisation of usual rewriting systems for handling non-terminating equations such as commutativity. They are studied in detail in Jouannaud (1983) and Peterson & Stickel (1981).

(E,R) is terminating iff there is no infinite sequence of E,R-reductions from any term.

(E,R) is inter-reduced iff for each rule $1 \to r$ in R, r is (E,R)-irreducible and 1 is $(E, (R\text{-}(1 \to r)))$-irreducible.

(E,R) is confluent iff for any terms t, $t1$ and $t2$, $t \overset{*}{\to} E,R\ t1$ and $t \overset{*}{\to} E,R\ t2$ implies: there exist $t3$ and $t4$ such that: $t1 \overset{*}{\to} E,R\ t3$, $t2 \overset{*}{\to} E,R\ t4$ and $t3 =E\ t4$.

(E,R) is canonical iff it is both terminating and confluent.

A term $t1$ is a normal form of $t2$ iff $t2 \overset{*}{\to} E,R\ t1$ and $t1$ is (E,R)-irreducible. It is easy to see that if (E,R) is canonical, then every term has a unique irreducible normal form (up to $=E$).

2.2. PROPOSITIONAL CALCULUS

Formulas of propositional calculus are constructed from the following vocabulary:

—two constants TRUE and FALSE.

—Boolean connectors:

$\vee$: or	$\langle=\rangle$: equivalence
$\&$: and	$=\rangle$: implication
$\neg$: not	$!$: exclusive or

—a denumerable set of Boolean variables, each variable ranging over the two values TRUE and FALSE.

To distinguish them from ordinary variables, Boolean variables will always be denoted by upper case letters: X, Y, Z or $X1, X2, \ldots, Xn$.

A formula is VALID or TAUTOLOGICAL iff its value is TRUE for each assignment of values to its variables, according to the rules of Boolean calculus. We embody these rules into the following ETRS:

—Equational system EBOOL:

$$X \vee Y = Y \vee X$$
$$(X \vee Y) \vee Z = X \vee (Y \vee Z)$$
$$X \,\&\, Y = Y \,\&\, X$$
$$(X \,\&\, Y) \,\&\, Z = X \,\&\, (Y \,\&\, Z)$$

—Rewriting system RBOOL:

(1) $X \langle=\rangle Y \to (X \vee (\neg Y)) \,\&\, (Y \vee (\neg X))$

(2) $X =\rangle Y \to (\neg X) \vee Y$

(3) $X\ !\ Y \to (X \vee Y) \,\&\, ((\neg X) \vee (\neg Y))$

(4) $\neg(\text{TRUE}) \to \text{FALSE}$

(5) $\neg(\text{FALSE}) \to \text{TRUE}$

(6) $\neg(X \vee Y) \to (\neg X) \,\&\, (\neg Y)$

(7) $\neg(X \,\&\, Y) \to (\neg X) \vee (\neg Y)$

(8) $\neg(\neg X) \to X$

(9) $X \vee (Y \,\&\, Z) \to (X \vee Y) \,\&\, (X \vee Z)$

(10) $X \vee \text{TRUE} \to \text{TRUE}$

(11) $X \vee \text{FALSE} \to X$

(12) $X \vee X \to X$

(13) $X \vee (\neg X) \to \text{TRUE}$

(14) $X \,\&\, \text{TRUE} \to X$

(15) $X \,\&\, \text{FALSE} \to \text{FALSE}$

(16) $X \& X \to X$

(17) $X \& (\neg X) \to \text{FALSE}$

Rules (1) to (3) eliminate the connectors $\langle = \rangle$, $=\rangle$, !. Rules (4) to (8) eliminate all the connectors $\neg$ which are not directly applied to variables. Rules (10), (11), (14), (15) eliminate the constants TRUE and FALSE inside other connectors. Rule (9) converts a formula into conjunctive normal form $C1 \& C2 \& \ldots \& Cm$, each Ci being of the form:

$$Ci = X1 \vee X2 \vee \ldots \vee Xp \vee (\neg Xp+1) \vee (\neg Xp+2) \vee \ldots \vee (\neg Xn)$$

For a given Ci, the Xjs are distinct variables according to rules (12) and (13).

Therefore, in this ETRS which we denote BOOL and which is obviously terminating, the normal forms of formulas are:

—The constants TRUE and FALSE.

—The formulas $C1 \& C2 \& \ldots \& Cm$, each Ci being a disjunction of variables and of negations of variables which are all distinct.

This system is not confluent on all formulas. Moreover, it is well known that it is impossible to construct a finite confluent rewriting system in propositional calculus based on connectors $\vee$, &, $\neg$, due to the fact that the prime implicant representation of Boolean formulas is not unique.

But this system is confluent on valid formulas. For if F were a valid formula and had the normal form $C1 \& C2 \& \ldots Cm$, with variables in each Ci distinct, it would be easy to assign values to the variables of F such that, for example, $C1 =$ FALSE and hence $F =$ FALSE. Therefore, the unique normal form of F is TRUE.

Given two formulas $F1$ and $F2$, we can decide whether $F1$ and $F2$ are equivalent by computing the normal form of $F1 \langle = \rangle F2$: $F1$ and $F2$ are equivalent iff this normal form is TRUE. For example, the absorption law (i.e. $X \& (X \vee Y) = X$) can be proved by reducing the formula $(X \& (X \vee Y)) \langle = \rangle X$:

$$(X \& (X \vee Y)) \langle = \rangle X \to ((X \& (X \vee Y)) \vee (\neg X)) \& (X \vee (\neg (X \& (X \vee Y)))) \text{ by rule (1)}$$
$$\to \ldots \to \text{TRUE by rules (4) to (17).}$$

Note that both formulas $X \& (X \vee Y)$ and X are irreducible.

Of course, this method is less efficient than using Hsiang's system, in which two formulas are equivalent iff they have the same normal form. But we shall retain this system for the moment, for it is easier to manipulate clauses with it than with Hsiang's system (we shall introduce Hsiang's system in Section 4).

Note that if we replace rule (9) by the other distributivity rule: $X \& (Y \vee Z) \to (X \& Y) \vee (X \& Z)$, we obtain a dual system confluent on unsatisfiable formulas (FALSE being the unique normal form of these formulas).

In the dual system, we can decide the equivalence of two formulas $F1$ and $F2$ by computing the normal form of $F1 \,!\, F2 : F1$ and $F2$ are equivalent iff this normal form is FALSE (recall that exclusive or is the negation of equivalence).

REMARK. Another system confluent on valid formulas has been found by Lankford & Musser (1978). This system is based on the If–Then–Else connector.

2.3. BIRKHOFF'S THEOREM, INDUCTIVE COMPLETENESS

We consider now BOOL as a set of non-oriented equations which defines an equational theory. We denote $=_{\text{BOOL}}$ the corresponding equality relation, and M(BOOL) the class of

all models of this theory; $M(\text{BOOL})$ is the family of Boolean algebras, whose initial model is the standard two-values model $B = \{\text{TRUE}, \text{FALSE}\}$.

It is well known (from Birkoff's theorem) that equational deduction is complete for $M(\text{BOOL})$:

$$M(\text{BOOL}) \models M = N \quad \text{iff} \quad M =_{\text{BOOL}} N$$

This property does not hold in general for proof theory in the initial model (see Huet & Oppen (1980): in this model, some kind of induction may be required. But this property holds for some theories, which are called *inductively complete* in a previous paper by the author (Paul, 1984).

More formally, for an equational theory T, we define two relations between terms: $=_T$ (equational equality) and $=_{IT}$ (inductive equality), as follows:

$M =_T N$ iff $M = N$ is valid in all models of T
$M =_{IT} N$ iff $M = N$ is valid in the initial model of T

Obviously, $=_T \subseteq =_{IT}$. T is inductively complete iff the converse is true, i.e. $=_{IT} \subseteq =_T$ (hence $=_{IT} = =_T$). (Another formulation is the following: T is inductively complete iff its initial model is "functionally free' for T, cf. Tarski (1946).)

THEOREM 2.1. BOOL *is inductively complete.*

PROOF. We will use in the proof the following properties of the connector $\langle = \rangle$, which can be proved by reducing the left-hand side and right-hand side of each equality to their normal form in the rewriting system BOOL:

Commutativity:	$X \langle = \rangle Y =_{\text{BOOL}} Y \langle = \rangle X$
Associativity:	$(X \langle = \rangle Y) \langle = \rangle Z =_{\text{BOOL}} X \langle = \rangle (Y \langle = \rangle Z)$
Identity:	$X \langle = \rangle \text{TRUE} =_{\text{BOOL}} X$
Nilpotence:	$X \langle = \rangle X =_{\text{BOOL}} \text{TRUE}.$

We denote the equality over the initial algebra B by $=_{\text{IBOOL}}$. Let F and G two formulas such that $F =_{\text{IBOOL}} G$. Using nilpotence, we can write: $F \langle = \rangle G =_{\text{IBOOL}} F \langle = \rangle F =_{\text{BOOL}} \text{TRUE}$.

Hence $F \langle = \rangle G =_{\text{IBOOL}} \text{TRUE}$, i.e. $F \langle = \rangle G$ is a tautology.

From the confluence of the rewriting system BOOL on tautologies, we obtain:

$$F \langle = \rangle G \xrightarrow{*}_{\text{BOOL}} \text{TRUE}, \quad \text{hence} \quad F \langle = \rangle G =_{\text{BOOL}} \text{TRUE}.$$

Then, using the properties of $\langle = \rangle$, we can write the sequence:

$$\begin{aligned} F &=_{\text{BOOL}} F \langle = \rangle \text{TRUE} =_{\text{BOOL}} F \langle = \rangle (F \langle = \rangle G) =_{\text{BOOL}} (F \langle = \rangle F) \langle = \rangle G \\ &=_{\text{BOOL}} \text{TRUE} \langle = \rangle G =_{\text{BOOL}} G \end{aligned}$$

Hence, $F =_{\text{BOOL}} G$, which ends the proof of Theorem 2.1.

We will show in Section 4 how the inductive completeness of BOOL can be extended from propositional calculus to quantifier-free predicate calculus.

It can also be shown that the system of Hsiang is inductively complete [see Paul (1984), where other examples of inductive completeness are given]. As a consequence, both systems, considered as equational theories, are equivalent. Actually, inductive

completeness of these two systems can be regarded merely as a new way of expressing the well-known completeness of propositional calculus.

3. Confluence on Valid Formulas in Predicate Calculus

3.1. REVIEW OF FIRST ORDER PREDICATE CALCULUS AND RESOLUTION

We start with a vocabulary of variables, function symbols, and predicate symbols. Terms, atoms and literals are introduced next [see for example Chang & Lee (1973) for a full description]. We do not consider here many-sorted predicate calculus.

Throughout this paper, we only deal with quantifier-free first order predicate calculus; i.e. all formulas manipulated are in prenex form with all their variables (implicitly) universally quantified.

A clause is a finite disjunction of zero or more literals; the order and parenthesis of the literals are irrelevant; in other words, a clause is assimilated to its EBOOL-equivalence class. The empty clause is assimilated to the constant FALSE. A tautological clause is a clause which contains a complementary pair of literals (i.e. L and $\neg L$).

A clause with repeated literals is considered as *different* from the same clause without repeated literals, because both clauses are not in the same EBOOL-equivalence class (note that they would be in the same EBOOL-equivalence class if the rule $X \vee X \rightarrow X$ of RBOOL were transferred into EBOOL, i.e. regarded as a non-oriented equation).

The resolution operation is decomposed into binary resolution and binary factoring, which are defined in the usual way. Deletion of repeated literals in a clause is considered as a particular case of factoring.

$|C|$ denotes the clause obtained from the clause C by retaining only one occurrence of repeated literals in C.

The clause $C1$ subsumes the clause $C2$ if there is a substitution σ and a sub-clause D of $C2$ such that $|C1\sigma| = D$.

A set of clauses is regarded as equivalent to the conjunction of these clauses. The union (i.e. conjunction) of two sets of clauses $S1$ and $S2$ is denoted $S1 \cup S2$.

If S is a set of clauses, the first order formula F is a valid consequence of S, or is valid in the theory specified by S, iff F is true in all models of S. In particular, TRUE is a valid consequence of any S, and FALSE is a valid consequence of S iff S is unsatisfiable.

We will use the following result obtained by Lee (1967):

THEOREM 3.1. (*Completeness of resolution for consequence finding.*)
Let S be a set of clauses.
Let C be a non-tautological clause.
C is a valid consequence of S iff there is a clause I deduced from S by resolution which subsumes C.

This theorem links the semantic concept of truth and the syntactic concept of provability (by resolution). It is proved in Lee (1967).

3.2. CONFLUENCE ON VALID FORMULAS

Let S be a set of clauses. We associate with S an equational term rewriting system which is the union of the two following systems:

—The system BOOL defined in Section 2.2.

—The set of rules $\{C \rightarrow \text{TRUE}\}$ with C in S.

We denote also S this ETRS.

We say that S is confluent on valid formulas iff for any quantifier-free formula F which is a valid consequence of S, $F \overset{*}{\rightarrow} S$ TRUE.

THEOREM 3.2. *Let S be a set of clauses which do not contain the empty clause. S is confluent on valid formulas iff the following conditions are met:*

(i) for each binary factor BF *of a clause in S,* BF $\overset{*}{\rightarrow} S$ TRUE;
(ii) for each binary resolvent BR *of two clauses in S,* BR $\overset{*}{\rightarrow} S$ TRUE.

This theorem corresponds to the Knuth & Bendix (1970) critical pairs theorem. In fact, the computation of binary factors and binary resolvents can be regarded as a computation of critical pairs between rules of S restricted to certain critical pairs (the details of this computation are given in the appendix). We do not need to compute all critical pairs because we do not require the general confluence, but only the confluence on valid formulas.

PROOF. The conditions (i) and (ii) are obviously necessary. Conversely, let us suppose that (i) and (ii) are true. To prove that S is confluent on valid formulas, we need the following lemmas.

LEMMA A. *If C is a non-tautological clause such that $C \overset{*}{\rightarrow} S$ TRUE, there is a clause D in S, a sub-clause B of C and a substitution σ such that $B = D\sigma$.*

PROOF. Obvious.

LEMMA B. *If C is a clause such that $C \overset{*}{\rightarrow} S$ TRUE, and* BF(C) *is a binary factor of C,* BF(C) $\overset{*}{\rightarrow} S$ TRUE.

PROOF. If C is tautological, BF(C) is also tautological. Hence, BF(C) $\overset{*}{\rightarrow}$BOOL TRUE and BF(C) $\overset{*}{\rightarrow} S$ TRUE since BOOL $\subseteq S$.

If C is not tautological, by using Lemma A, C can be written:

$$C = D\sigma \vee C1, \quad \text{with} \quad D \text{ in } S.$$

If the factoring of C is done between two literals of $C1$, or between a literal in $C1$ and a literal in $D\sigma$, it is easy to check that BF(C) $\overset{*}{\rightarrow} S$ TRUE by application of the rule $D \rightarrow$ TRUE.

If the factoring is done between two literals $L1\sigma$ and $L2\sigma$ of $D\sigma$, we have:

$$C = L1\sigma \vee L2\sigma \vee D1\sigma \vee C1, \quad \text{with} \quad D = L1 \vee L2 \vee D1.$$

Let θ be the mgu of $L1\sigma$ and $L2\sigma$. We have:

$$\text{BF}(C) = L1\sigma\theta \vee D1\sigma\theta \vee C1\theta.$$

$L1$ and $L2$ are unifiable, since $L1\sigma$ and $L2\sigma$ are unifiable. Let ψ be the mgu of $L1$ and $L2$: $\text{BF}(D) = L1\psi \vee D1\psi$ is a binary factor of D. By hypothesis (i), BF(D) $\overset{*}{\rightarrow} S$ TRUE. From the definition of the mgu, there is a substitution ω such that $\sigma\theta = \psi\omega$. Hence,

$$\text{BF}(C) = \text{BF}(D)\omega \vee C1\theta \quad \text{and} \quad \text{BF}(C) \overset{*}{\rightarrow} S \text{ TRUE}.$$

LEMMA C. *If C and D are two clauses such that $C \overset{*}{\rightarrow} S$ TRUE and $D \overset{*}{\rightarrow} S$ TRUE,* and BR(C, D) *is a binary resolvent of C and D,* BR(C, D) $\overset{*}{\rightarrow} S$ TRUE.

PROOF. Similar to the proof of Lemma B.

We can now prove that S is confluent on valid formulas: let F be a formula which is a valid consequence of S.

If F is a tautology, $F \overset{*}{\to}_{\text{BOOL}} \text{TRUE}$ and hence: $F \overset{*}{\to}_S \text{TRUE}$.

If F is not a tautology, $F \overset{*}{\to}_{\text{BOOL}} C1 \,\&\, \ldots \,\&\, Cp$, each Ci being a non-tautological clause which is also a valid consequence of S.

Let us choose, for example, the clause $C1$: by Theorem 3.1, there is a clause I deduced from S by resolution which subsumes $C1$. Therefore, there is a subclause D of $C1$ and a substitution σ such that $I\sigma$ and D contain exactly the same literals (possibly repeated for $I\sigma$, but not for D because D is reduced by the system BOOL).

I is deduced from S by resolution, i.e. by a sequence of binary resolution and binary factoring. Hence, from Lemmas B and C: $I \overset{*}{\to}_S \text{TRUE}$. Therefore, $I\sigma \overset{*}{\to}_S \text{TRUE}$.

D is obtained from $I\sigma$ by eventual deletion of repeated literals, which is a particular case of factoring. Hence, by Lemma B, $D \overset{*}{\to}_S \text{TRUE}$.

Since S does not contain the empty clause, D is not the empty clause.

Since D is a non-empty subclause of $C1$, $C1 \overset{*}{\to}_S \text{TRUE}$.

We prove in the same way that $Ci \overset{*}{\to}_S \text{TRUE}$ for all i. Therefore, $F \overset{*}{\to}_S \text{TRUE}$, which ends the proof of Theorem 3.2.

Note that we have also proved that S is satisfiable, since we do not have $\text{FALSE} \overset{*}{\to}_S \text{TRUE}$; hence, FALSE is not a valid consequence of S.

3.3 COMPLETION ALGORITHM

From Theorem 3.2, we can derive a completion algorithm similar to the Knuth & Bendix algorithm, which will generate from any set of clauses an equivalent (i.e. having the same models) rewriting system confluent on valid formulas.

In the following, Ei and Ri are two finite sets of clauses. The current rewriting system is the system associated with Ri, i.e. the system: $\text{BOOL} \cup \{C \to \text{TRUE}, C \text{ in } Ri\}$. This rewriting system is also denoted Ri.

Each clause in Ri has a label, which is a unique integer. We denote by $k : C$ the clause C with label k. Finally, each clause in Ri is marked or unmarked.

Initial data: a (finite) set of clauses S.

1. Initialisation: Let $E0 = S$, $R0 = \text{BOOL}$, $i = 0$, $p = 0$.
2. If $Ei \neq$ emptyset, go to 4.
3. Compute binary resolvents and binary factors:

If all clauses in Ri are marked, stop with success. Otherwise, select an unmarked clause C in Ri, say with label k. Do:

$Ei+1 = \{\text{binary factors of } C\} \cup \{\text{binary resolvents of } (C, D) \text{ for any clause } D \text{ of } Ri \text{ of label not greater than } k\}$

$Ri+1 = Ri$ with clause C marked
$i = i+1$
go to 2.

4. Introduction of new rules:

Select clause C in Ei. Let $C!$ a Ri-normal form of C. From the structure of Ri, it is easy to see that $C!$ can be either a (non-empty) clause, or one of the constants TRUE or FALSE.

—If $C! = \text{FALSE}$, stop with answer: S unsatisfiable.
—If $C! = \text{TRUE}$, do: $Ei+1 = Ei-C$, $Ri+1 = Ri$, $i = i+1$ and go to 2.
—If $C! \# \text{FALSE}$ and $C! \# \text{TRUE}$:

Let K be the set of labels of clauses in Ri reducible by the new rule $C! \to \text{TRUE}$. Do:

$Ei+1 = (Ei-C) \cup \{j : D \text{ with } j \text{ in } K\}$
$p = p+1$
$Ri+1 = \{j : D \text{ with } D \text{ in } Ri \text{ and } j \text{ not in } K\} \cup \{p : C!\}$
$i = i+1$
go to 2.

In $Ri+1$, the clauses coming from Ri are marked or unmarked as they were in Ri, the new clause $p : C!$ is unmarked.

To ensure the completeness of this algorithm, we need an assumption concerning the selection of clause at step 3: for every clause label k, there is an iteration i such that either the clause of label k is deleted from Ri (i.e. k is in K at iteration i), or the clause of label k is selected at step 3. This assumption ensures that no clause will be ignored indefinitely by the selection process.

This algorithm is basically the same as breadth-first resolution. The deletion at step 4 of clauses which are reduced to TRUE corresponds to the deletion, in resolution algorithm, of clauses which are tautologies or which are subsumed by other clauses (however, note that the inference of the rule $C \to \text{TRUE}$ from the rules $C \vee \neg L \to \text{TRUE}$ and $L \to \text{TRUE}$, L being a positive literal, is performed in this algorithm by interreduction of rules and not by resolution).

On the other hand, if we compare this algorithm with the Knuth & Bendix completion algorithm, as presented for instance in Huet (1981), the only difference is the replacement of the computation of critical pairs by the computation of binary factors and binary resolvents, i.e. of only certain critical pairs. Consequently, there is no case of stop with failure, because the right-hand side of the added rules is always TRUE. The algorithm may:

—either stop with the conclusion: S is unsatisfiable,
—either stop with success,
—or run forever.

In the last two cases, let $R\infty$ be the final rewriting system constructed by the algorithm. $R\infty$ is the set of all the rules which belong to some Ri and to all Rjs for $j > i$; i.e. which are never reduced by other rules. In the first case, we define $R\infty$ as the single rule $X \to \text{TRUE}$. Note that this rule is obtained by superposing the rule $\text{FALSE} \to \text{TRUE}$, generated by the algorithm, and the rule $X \vee \text{FALSE} \to X$ of BOOL; furthermore, this rule entails the deletion of all previous rules by redundancy (including rules of BOOL).

THEOREM 3.3. *The rewriting system $R\infty$ has the following properties:*

(*i*) *$R\infty$ is inter-reduced.*
(*ii*) *$R\infty$ (considered as a set of clauses) is equivalent to the initial set of clauses S.*
(*iii*) *$R\infty$ is confluent on valid formulas.*

Furthermore, for a given S, $R\infty$ is the only rewriting system associated with a set of clauses which has these properties.

PROOF. If S is unsatisfiable, $R\infty$ is reduced to the only rule $X \to \text{TRUE}$, which means that

any formula can be considered as a valid consequence of S. Properties (i) to (iii) are obvious.

If S is satisfiable, (i) comes from the structure of the algorithm. (ii) comes from the fact that at each iteration i of the algorithm, S is equivalent to $Ei \cup Ri$. Finally, (iii) comes from Theorem 3.2.

To prove the unicity, let us suppose that there exists another rewriting system Q, associated with a set of clauses, such that Q has the properties (i) to (iii). As Q is equivalent to S by (ii), Q is equivalent to $R\infty$.

Let $C1$ be a clause in Q. $C1$ is a valid consequence of $R\infty$, and since $R\infty$ is confluent on valid formulas, $C1 \xrightarrow{*} R\infty$ TRUE.

$C1$ is not tautological (otherwise, it could be reduced by rules of BOOL, and the system Q would not be inter-reduced). Therefore, by Lemma A, there is a clause $C2$ in $R\infty$ and a substitution σ such that $C2\sigma \subseteq C1$. This clause $C2$ is a valid consequence of Q, and we can prove in the same way that there is a clause $D1$ in Q and a substitution θ such that $D1\theta \subseteq C2$.

Hence $D1\theta\sigma \subseteq C1$. Since Q is interreduced, that entails $D1 = C1$, hence $\theta\sigma =$ identity and $C1$ and $C2$ are identical (up to the names of variables). Therefore $Q \subseteq R\infty$. We prove in the same way that $R\infty \subseteq Q$, therefore $Q = R\infty$.

Note the difference with the Knuth & Bendix algorithm, in which it is possible to generate several different confluent rewriting systems, depending on the orientation chosen for rewrite rules.

4. Equational Methods in First Order Predicate Calculus

4.1. BIRKHOFF'S THEOREM

Let $S = \{E1, \ldots, En\}$ be a theory defined by a set of quantifier-free formulas. $E1, \ldots, En$ are not necessarily in clausal form.

We associate with S an equational system which is the union of the following systems:

—The system BOOL defined in Section 2.2 (regarded as a set of equations).
—The set of equations $\{Ei = \text{TRUE}\}$ with Ei in S.

We denote also S this equational system.

Note that the symbol $=$, linking two Boolean formulas, corresponds here to the Boolean connector $\langle=\rangle$ and is different from the equality predicate, linking two non-Boolean terms, which can also exist in S.

This equational system defines an equality relation on the set of quantifier-free formulas. We denote this relation $=_S$.

THEOREM 4.1. *Let F and G be two quantifier-free formulas. F and G have simultaneously the same values* (TRUE *or* FALSE) *for every assignment of their variables in all the models of S iff $F =_S G$. An equivalent statement is:*

$$S \models F \langle=\rangle G \text{ iff } F =_S G.$$

Theorem 4.1 states that equational-like deduction is complete for quantifier-free first order theories. The Boolean connector $\langle=\rangle$ plays exactly the same role as the equality predicate in the usual Birkhoff theorem for equational theories.

To prove Theorem 4.1, we need the following lemmas:

LEMMA D. *For every set of formulas S, there is an equivalent set of clauses S1 such that:* $=S= =S1$.

PROOF. By using BOOL, each Ei in S can be transformed into a conjunction of clauses. Let $S1$ be the set of clauses obtained in this way. For each Ei in S, we have the following relation:

$$Ei =_{\text{BOOL}} C1 \,\&\, C2 \,\&\, \ldots \,\&\, Cp \text{ with } Cj \text{ in } S1.$$

Since BOOL $\subseteq S1$, we have $Ei =_{S1} \text{TRUE}$. Hence $=S \subseteq =S1$. Conversely, the following sequence holds for each Cj:

$$\begin{aligned} Cj &=_{\text{BOOL}} Cj \,\&\, \text{TRUE} =_S Cj \,\&\, Ei =_{\text{BOOL}} Cj \,\&\, (C1 \,\&\, \ldots \,\&\, Cp) \\ &=_{\text{BOOL}} C1 \,\&\, \ldots \,\&\, Cp =_{\text{BOOL}} Ei =_S \text{TRUE}. \end{aligned}$$

Therefore $Cj =_S \text{TRUE}$. Hence $=S1 \subseteq =S$.

LEMMA E. *Let S be a set of clauses, and R∞ the rewriting system built from S by the completion algorithm. We have* $=S= =R\infty$.

PROOF. If C is a clause in S, $C \overset{*}{\to} R\infty$ TRUE. Then $=S \subseteq =R\infty$. Conversely, if C is a clause in $R\infty$, C is obtained from the clauses in S by a finite sequence of binary factoring and binary resolution. These operations are particular cases of computation of critical pairs (see appendix). Consequently, the equation $C =$ TRUE is an equational consequence of S; therefore $=R\infty \subseteq =S$.

We can now prove Theorem 4.1:

Let F and G be two quantifier-free formulas. If $F =_S G$, since the rule of substitution of equivalents is valid in first order predicate calculus, $F \langle=\rangle G$ is a valid consequence of S.

Conversely, let us suppose that $F \langle=\rangle G$ is a valid consequence of S. From Lemma D, we can suppose that S is clausal. Let $R\infty$ be the rewriting system obtained from S by the completion algorithm. We have:

$$F \langle=\rangle G \overset{*}{\to} R\infty \text{ TRUE}.$$

Hence, $F \langle=\rangle G =_S \text{TRUE}$ by Lemma E. Then, using the properties of the connector $\langle=\rangle$ (see Section 2.3), we can write the following sequence:

$$\begin{aligned} F &=_{\text{BOOL}} F \langle=\rangle \text{TRUE} =_S F \langle=\rangle (F \langle=\rangle G) =_{\text{BOOL}} (F \langle=\rangle F) \langle=\rangle G \\ &=_{\text{BOOL}} \text{TRUE} \langle=\rangle G =_{\text{BOOL}} G \end{aligned}$$

Hence, $F =_S G$, which ends the proof.

REMARK. Theorem 4.1 has the following particular cases:

—With $G =$ TRUE: F is a valid consequence of S iff $F =_S \text{TRUE}$.

—With $F =$ FALSE and $G =$ TRUE: FALSE is a valid consequence of S (i.e. S is unsatisfiable) iff FALSE $=_S$ TRUE.

4.2. COMPARISON WITH THE USUAL BIRKHOFF THEOREM

Consider a two-sorted equational theory Q with a sort Bool, defined by the same set of equations as S at the beginning of Section 4.1. Note that in Q, predicate symbols are only function symbols with target values of sort Bool.

Birkhoff's theorem extended to many-sorted equational theories states that two

expressions are equal in all the models of the theory Q iff they can be proved equal by pure equational deduction (the extension to the many-sorted case can be done if we consider only models in which there are no empty sorts: cf. Huet & Oppen (1980).

This result is identical to the result of Theorem 4.1, but there is a major *semantic* difference: we need to consider all the models of the equational theory Q. In some of these models, the carrier of the sort Bool can be different from the initial two-values model {TRUE, FALSE}. Actually, the carrier of sort Bool can be any Boolean algebra.

Theorem 4.1 is much stronger, for it proves that equational deduction is still complete if we restrict ourselves to the models whose carrier of sort Bool is the initial model; for these models are the only ones to be considered in first order predicate calculus. These models are indeed the most interesting in practice.

Actually, Theorem 4.1 is an extension of the property of "inductive completeness" of the system BOOL, defined in Section 2.3, to quantifier-free predicate calculus.

REMARK. Theorem 4.1 remains true if BOOL is replaced by any equational specification BOOL1 such that $=_{\text{BOOL}} = =_{\text{BOOL1}}$ (i.e. any inductively complete specification of propositional calculus). In particular, Theorem 4.1 is true if BOOL1 is the Hsiang system.

4.3. GENERAL CONFLUENCE IN FIRST ORDER PREDICATE CALCULUS

Theorem 4.1 states that equational deduction is complete for quantifier-free first order theories. Consequently, we can try to build from any theory S a rewriting system confluent on all formulas and not only on valid formulas, by using the Knuth & Bendix algorithm instead of the resolution algorithm (i.e. by computing all critical pairs and not only certain critical pairs). But we will not succeed if we start with the rewriting system BOOL, because even if we restrict ourselves to propositional calculus, it is impossible to build from BOOL a finite complete rewriting system.

Therefore, we must use for this purpose Hsiang's system.

4.3.1. HSIANG'S SYSTEM AND THE DUAL SYSTEM

The Hsiang system is the following:

(1) $X \,\&\, X \to X$

(2) $X \,\&\, \text{TRUE} \to X$

(3) $X \,\&\, \text{FALSE} \to \text{FALSE}$

(4) $X \,!\, \text{FALSE} \to X$

(5) $X \,!\, X \to \text{FALSE}$

(6) $X \,\&\, (Y \,!\, Z) \to (X \,\&\, Y) \,!\, (X \,\&\, Z)$

(7) $X \langle = \rangle Y \to X \,!\, Y \,!\, \text{TRUE}$

(8) $\neg X \to X \,!\, \text{TRUE}$

(9) $X \vee Y \to (X \,\&\, Y) \,!\, X \,!\, Y$

(10) $X \Rightarrow Y \to (X \,\&\, Y) \,!\, X \,!\, \text{TRUE}$

This system is based on the connectors & and !. Rules (7) to (10) eliminate other connectors. This system is confluent modulo the associativity/commutativity of & and !.

As noted by Hsiang, there exists also a dual system. To formalise the construction of this system, we define the dual $d(f)$ of a Boolean function f with arity n by:

$$d(f)(X1, X2, \ldots, Xn) = \neg (f(\neg X1, \neg X2, \ldots, \neg Xn)).$$

This definition is extended to any Boolean formula, by considering it as a function of its variables.

The following relations are straightforward:

$$d(\text{TRUE}) = \text{FALSE} \qquad d(\text{FALSE}) = \text{TRUE}$$
$$d(\vee) = \& \qquad d(\&) = \vee$$
$$d(\langle = \rangle) = ! \qquad d(!) = \langle = \rangle$$
$$\text{(In general, } d(d(f)) = f\text{)}$$
$$d(\neg) = \neg$$
$$d(X) = X \text{ for any Boolean variable X.}$$

$d(f(P1, \ldots, Pn)) = d(f)(d(P1), \ldots, d(Pn)$ for any Boolean function f and Boolean formulas $P1, \ldots, Pn$.

These relations allow us to compute the dual of any formula, constructed from Boolean variables and connectors, by induction over the structure of this formula.

If $P1$ and $P2$ are two equivalent formulas, $d(P1)$ and $d(P2)$ are also equivalent. This property allows us to apply a transformation by duality to Hsiang's system. We obtain mechanically the following system:

(1) $X \vee X \to X$

(2) $X \vee \text{FALSE} \to X$

(3) $X \vee \text{TRUE} \to \text{TRUE}$

(4) $X \langle = \rangle \text{TRUE} \to X$

(5) $X \langle = \rangle X \to \text{TRUE}$

(6) $X \vee (Y \langle = \rangle Z) \to (X \vee Y) \langle = \rangle (X \vee Z)$

(7) $X ! Y \to X \langle = \rangle Y \langle = \rangle \text{FALSE}$

(8) $\neg X \to X \langle = \rangle \text{FALSE}$

(9) $X \& Y \to (X \vee Y) \langle = \rangle X \langle = \rangle Y$

(10) $X = \rangle Y \to (X \vee Y) \langle = \rangle Y$

(The rule (10) is directly computed, since the dual of $= \rangle$ is not a usual connector.)

This system is built from $\vee$ and $\langle = \rangle$ instead of & and !. It is confluent modulo the associativity/commutativity of these two connectors.

4.3.2. COMPLETION ALGORITHM

LET $S = \{E1, \ldots, En\}$ be a theory defined by a set of quantifier-free formulas.

To complete S, we run the Knuth & Bendix algorithm in its associative/commutative version (Peterson & Stickel, 1981), initialising the set of rewrite rules with the Hsiang system (or the dual system) and the set of equations with $\{Ei = \text{TRUE}\}$. If a formula Ei is already in the form $Fi \langle = \rangle Gi$, we can initialise the set of equations with the equation $Fi = Gi$ instead of $(Fi \langle = \rangle Gi) = \text{TRUE}$.

From now on, we suppose that the completion algorithm does not stop with failure because of the generation of a non-orientable equation. Let $R\infty$ be the (finite or infinite) rewriting system built by the algorithm.

THEOREM 4.2. *Let F and G be two quantifier-free formulas. The formula $F \langle = \rangle G$ is a valid consequence of S iff F and G have the same $R\infty$ normal form. In particular, F is a valid consequence of S iff $F \overset{*}{\to} R\infty$ TRUE.*

PROOF. From Theorem 4.1 and the general properties of the Knuth & Bendix algorithm, as explained in Huet (1981).

EXAMPLE. Let us consider the following clausal specification:

$$S = \{P(f(x)) \vee P(x), \quad \neg P(f(x)) \vee \neg P(x)\}.$$

If we apply to this specification the completion algorithm of Section 3.3, based on resolution, we generate an infinite number of rewrite rules:

(system BOOL+)
$P(f(x)) \vee P(x) \rightarrow$ TRUE
$\neg P(f(x)) \vee \neg P(x) \rightarrow$ TRUE
$\neg P(f(f(x))) \vee P(x) \rightarrow$ TRUE
$P(f(f(f(x)))) \vee P(x) \rightarrow$ TRUE
$\neg P(f(f(f(f(x))))) \vee P(x) \rightarrow$ TRUE
...

But if we apply to this specification the Knuth & Bendix completion algorithm, this algorithm will stop with only a finite number of rules, namely:

(Hsiang's system+)
$P(f(x)) \rightarrow P(x)$! TRUE.

This rule expresses that $P(f(x))$ is the negation of $P(x)$.

Let us consider another example: let S be the following specification for a program to test if an element u is a member of a sequence z:

(1) $x = x$
(2) $\neg$ Elem(u, NIL)
(3) Elem(u, w . x) $\langle=\rangle$ ($u = w$) $\vee$ Elem(u, x)

We have added (1), simple reflexivity, the only equality property needed here. Using the dual Hsiang system and the Knuth & Bendix algorithm, we obtained at once the following complete system:

(dual Hsiang's system+)
$x = x \rightarrow$ TRUE
Elem(u, NIL) $\rightarrow$ FALSE
Elem(u, w . x) $\rightarrow$ ($u = w$) $\vee$ Elem(u, x).

If we use the resolution algorithm, we must split the equivalence into three clauses. We obtain an infinite system:

(SYSTEM BOOL+)
$x = x \rightarrow$ TRUE
$\neg$ Elem(u, NIL) $\rightarrow$ TRUE
$\neg$ Elem(u, w . x) $\vee$ ($u = w$) $\vee$ Elem(u, x) $\rightarrow$ TRUE
$\neg$ ($u = w$) $\vee$ Elem(u, w . x) $\rightarrow$ TRUE
$\neg$ Elem(u, x) $\vee$ Elem(u, w . x) $\rightarrow$ TRUE
$\neg$ Elem(u, w . NIL) $\vee$ ($u = w$) $\rightarrow$ TRUE
Elem(u, u . x) $\rightarrow$ TRUE

$$\neg(u = w) \vee \text{Elem}(u, w1.(w.x)) \rightarrow \text{TRUE}$$
$$\text{Elem}(u, w1.(u.x)) \rightarrow \text{TRUE}$$
$$\ldots$$

In both examples, the Knuth & Bendix algorithm is preferable to the resolution algorithm. That is due to the fact that it can discover and use as simplifiers interesting equivalence relations.

In other cases, both algorithms will run in parallel. For example, if S is $\{\neg P(x) \vee P(f(x))\}$, each algorithm generates infinitely many rules. Each rule:

$$\neg P(x) \vee P(f(f(f(\ldots(x))))) \rightarrow \text{TRUE}$$

produced by the resolution algorithm is associated with the rule:

$$P(x) \,\&\, P(f(f(f(\ldots(x))))) \rightarrow P(x)$$

produced by the Knuth & Bendix algorithm.

4.3.3. KNUTH–BENDIX ALGORITHM AS A REFUTATIONAL PROOF TECHNIQUE

Theorem 4.2 can be particularised for unsatisfiable specifications as follows:

THEOREM 4.3. *Let S be a set of quantifier-free formulas. S is unsatisfiable iff the Knuth & Bendix completion algorithm generates one of the following rules (X being a Boolean variable):*

$$X \rightarrow \text{TRUE} \qquad X \rightarrow \text{FALSE}$$
$$\text{FALSE} \rightarrow \text{TRUE} \qquad \text{TRUE} \rightarrow \text{FALSE}$$

PROOF. Taking $F = \text{FALSE}$ and $G = \text{TRUE}$ in Theorem 4.2, we obtain: FALSE is a valid consequence of S (i.e. S is unsatisfiable) iff FALSE and TRUE have the same normal form in $R\infty$. Therefore, either FALSE or TRUE must be reducible by $R\infty$. Hence, one of the above listed rules has been generated by the algorithm.

Note that in any case, the final rewriting system will be reduced to the only rule $X \rightarrow \text{TRUE}$ or $X \rightarrow \text{FALSE}$, all other rules being eliminated by redundancy.

Theorem 4.3 is close to results of Hsiang (1983) and Fages (1983).

4.3.4. FAILURE CASES OF THE ALGORITHM

If the algorithm stops with failure, we can in certain cases run it again after putting the incomparable critical pair into the set of non-oriented equations, with the associativity/commutativity of ! and & (or $\langle=\rangle$ and $\vee$ if we use the dual system). This method can be applied if we know a unification algorithm for the set of non-oriented equations (Jouannaud, 1983); for example, commutative predicates can be handled in this way.

As an example in which this method does not apply, we run the completion algorithm with S being the axiomatisation of equality (reflexivity, symmetry, transitivity, substitution); we put the commutativity rule $(x = y) = (y = x)$ into the set of non-oriented equations, but we generate at once another non-orientable critical pair: $(x = y)$ & $(x = z) = (x = y)$ & $(y = z)$, which cannot be handled by the unification techniques presently known.

Another obvious method for handling a non-orientable equation $F = G$ would be to

replace it by the equation $F\,!\,G = \text{FALSE}$ (or $F \langle = \rangle G = \text{TRUE}$ if we use the dual system). Such a modification of the completion procedure will delete the property of general confluence of $R\infty$, but we conjecture that a partial completeness property is preserved, such as:

(i) $R\infty$ is confluent on valid formulas (consequence-finding completeness).
(ii) If S is unsatisfiable, one of the rules listed in Theorem 4.3 is eventually generated (refutation completeness).

Note that (i) is stronger than (ii).

5. First Order Predicate Calculus in an Equational Theory

5.1. PRELIMINARIES

We are now in first order predicate calculus with equality. We consider a set of clauses which is the union of two subsets of the following form:

—A set of T of unit clauses of the form $\{M = N\}$ which define an equational theory.
—A set S of clauses which do not contain any equality predicate.

We suppose that the equational theory T can be compiled into a canonical term rewriting system R. We introduce a new inference rule, called *narrowing* defined as follows:

Given a clause C, if there is a non-variable occurrence u in C such that C/u is unifiable with the left-hand side of a rule $(1 \to r)$ in R with mgu σ, the clause $N(C) = C(u <- r)\sigma$ is a narrowing of C.

This is Hullot's definition, and not Lankford's or Slagle's, for we do not normalize $N(C)$ by R. Note that the pair $\langle N(C), \text{TRUE} \rangle$ is a critical pair between the two rules $1 \to r$ and $C \to \text{TRUE}$.

A resolution of the clauses $C1$ and $C2$ producing the clause $C3$ is said to be a *blocked* resolution if the three clauses $C1, C2, C3$ are in R-normal form. We similarly define a blocked binary resolution and a blocked binary factoring.

We need the following result, which is analogous to Theorem 3.1 (Lee's theorem):

THEOREM 5.1. *Let C be a non-tautological clause in R-normal form, which does not contain any equality predicate. C is a valid consequence of $T \cup S$ in predicate calculus with equality iff there is a clause I deduced from S by narrowing and blocked resolution which subsumes C.*

PROOF. We first consider the case where the clause C and all clauses in S are ground. If $S!$ is the set of R-normal forms of clauses in S, $S!$ is deduced from S by reduction by R, which is a particular case of narrowing; and C is a valid consequence of $S! \cup T$ in predicate calculus with equality. Therefore, $S! \cup T \cup \neg C$ is equality unsatisfiable.

Since $S!$ and $\neg C$ are in R-normal form, by using methods of Lankford (1975) we obtain that $S! \cup \neg C$ is unsatisfiable. Therefore, C is a valid consequence of $S!$.

From Lee's theorem, there is a clause I deduced from $S!$ by resolution which subsumes C. Moreover, this deduction is blocked.

This proof is easily lifted to the general case by using methods of Lee (1967), the usual lifting lemma for resolution, and the following lifting lemma for narrowing:

LEMMA F. *Let C be a clause. Let σ be a R-normalised substitution (i.e. $X\sigma$ is R-irreducible for each variable X). Let D be the R-normal form of $C\sigma$. There exists a clause $C1$ derived from C by a finite sequence of narrowings and a substitution θ such that $C1\theta = D$.*

PROOF. See Hullot (1980) [Lankford (1975) proves an analogous theorem when narrowing comprises a normalisation by R].

5.2. CONFLUENCE ON VALID FORMULAS

We associate with $T \cup S$ an equational term rewriting system which is defined as the union of the three following systems:

—The system BOOL defined in Section 2.2.
—The rewriting system R associated with the equational theory T.
—The set of rules $\{C \rightarrow \text{TRUE}\}$ with C in S.

We denote RS this equational term rewriting system.

We say that RS is confluent on valid formulas iff for each formula F without equality predicate which is a valid consequence of $T \cup S$ in predicate calculus with equality, $F \overset{*}{\rightarrow} RS$ TRUE.

THEOREM 5.2. *Let $T \cup S$ be defined as above. We suppose that S does not contain the empty clause. The ETRS RS is confluent on valid formulas iff the following conditions are met:*

(i) for each binary factor BF of a clause in S, $BF \overset{*}{\rightarrow} RS$ TRUE;
(ii) for each binary resolvent BR of two clauses in S, $BR \overset{*}{\rightarrow} RS$ TRUE;
(iii) for each narrowing N of a clause in S, $N \overset{*}{\rightarrow} RS$ TRUE.

PROOF. The conditions (i), (ii), (iii) are obviously necessary. Conversely, let us suppose that (i), (ii), (iii) are true. The run of the proof follows closely the proof of Theorem 3.2. Lemmas A, B, C have to be proved only for clauses in R-normal form (that is sufficient because we use only blocked resolution in Theorem 5.1). This fact allows us to ignore the rewriting system R; consequently, the proofs of these lemmas are exactly the same as for Theorem 3.2.

We need the following additional lemma, which extends Lemmas B and C to the narrowing operation:

LEMMA G. *If C is a clause such that $C \overset{*}{\rightarrow} RS$* TRUE, *and $N(C)$ is a narrowing of C, $N(C) \overset{*}{\rightarrow} RS$* TRUE.

PROOF. Let $RS1$ be the equational term rewriting system obtained from RS by retaining only the following rules of the system BOOL:

$$\neg(\text{TRUE}) \rightarrow \text{FALSE}$$
$$\neg(\text{FALSE}) \rightarrow \text{TRUE}$$
$$X \vee \text{TRUE} \rightarrow \text{TRUE}$$
$$X \vee \text{FALSE} \rightarrow X$$
$$X \vee X \rightarrow X$$
$$X \vee (\neg X) \rightarrow \text{TRUE}$$

We are going to prove that $RS1$ is confluent by using the confluence criterion of Peterson & Stickel. This criterion consists in checking the confluence of all AC-critical

pairs between rules of $RS1$ and their extensions. AC-critical pairs means that we use associative/commutative unification, $\vee$ being the symbol declared associative/commutative (see Peterson & Stickel (1981) for more details).

A simple case analysis shows that all AC-critical pairs between rules of $RS1$ are confluent:

—Critical pairs between rules of R are reduced because R is confluent.
—Critical pairs between rules of R and rule $D \rightarrow$ TRUE, D in S, are confluent from hypothesis (iii), because such critical pairs correspond to a narrowing of D.
—Critical pairs between rule $D \rightarrow$ TRUE, D in S, and rule $X \vee X \rightarrow X$ of BOOL (or its extension $Y \vee X \vee X \rightarrow Y \vee X$) are confluent from hypothesis (i), because such critical pairs correspond to a binary factoring of D.
—Critical pairs between rule $D1 \rightarrow$ TRUE and rule $D2 \rightarrow$ TRUE, $D1$ and $D2$ in S, with:

$$D1 = D3 \vee \neg L1$$
$$D2 = L2$$

$L1$ and $L2$ being two positive unifiable literals, are confluent from hypothesis (ii), because such critical pairs correspond to a binary resolution between $D1$ and $D2$.

—Other critical pairs are obviously confluent (in particular, the sub-system of BOOL which we use is confluent).

Let C be a clause such that $C \overset{*}{\rightarrow} RS$ TRUE and $N(C)$ a narrowing of C. We have $C \overset{*}{\rightarrow} RS1$ TRUE because the rules of BOOL which are not in $RS1$ can never be applied when the formula to reduce is a clause. From $C \overset{*}{\rightarrow} RS1$ TRUE, we have $C = RS1$ TRUE.

Hence $N(C) = RS1$ TRUE since $R \subseteq RS1$, and from the definition of narrowing.

Hence $N(C) \overset{*}{\rightarrow} RS1$ TRUE by confluence of $RS1$.

Hence $N(C) \overset{*}{\rightarrow} RS$ TRUE since $RS1 \subseteq RS$, which ends the proof of Lemma G.

We can now prove that the system RS is confluent on valid formulas by using Theorem 5.1 which characterises the valid formulas. The proof is exactly the same as the end of the proof of Theorem 3.2 and so is omitted.

From Theorem 5.2, we deduce a completion algorithm which is similar to the completion algorithm of Section 3.3. The differences are:

—We initialise the set of rules $R0$ to BOOL $\cup R$.
—We add at step 3 the computation of narrowings.

If $R\infty$ is the final rewriting system produced by the algorithm, we have the theorem:

THEOREM 5.3. *The rewriting system $R\infty$ has the following properties:*

(*i*) *$R\infty$ is interreduced*
(*ii*) *$R\infty$ is equivalent to $S \cup T$*
(*iii*) *$R\infty$ is confluent on valid formulas.*

Furthermore, for a given rewriting system R associated with the equational theory T, $R\infty$ is the only rewriting system associated with a set of clauses which has these properties.

PROOF. This proof follows closely the proof of Theorem 3.3 and is left to the reader.

5.3. EXTENSION TO EQUATIONAL TERM REWRITING SYSTEM

We suppose now that the equational theory T can be compiled into a canonical equational term rewriting system (P, R), P being a set of equations and R being a set of rewrite rules, as described in Section 2.1.

We suppose that there is a finite and complete algorithm of P-unification. We define P-resolution, P-binary resolution, P-binary factoring, P-narrowing as resolution, ... in which P-unification is used instead of ordinary unification. We define also P-subsumption as subsumption in which P-matching is used instead of matching.

To extend the previous results, we need an additional property of the ETRS (P, R). This property has been introduced by Jouannaud (1983) and is called P-coherence. Roughly speaking, P-coherence allows us to replace the reduction relation $\rightarrow P, R$ defined in Section 2.1 by a weaker relation $\rightarrow R, P$ defined as follows:

The term $t1$ R, P-reduces at occurrence u to a term $t2$ using the rule $1 \rightarrow r$ in R iff there exists a substitution σ such that $t1/u =_P 1\sigma$ and $t2 = t1[u <- r\sigma]$. Note that R, P-reduction is the same as R-reduction except that we use P-matching instead of matching.

Jouannaud gives sufficient conditions for testing simultaneously confluence and P-coherence of an ETRS. See Jouannaud (1983) for details. These results extend the previous results of Peterson & Stickel.

For our purpose, P-coherence allows to generalise Lemma F (lifting lemma for narrowing) if we use P-narrowing instead of narrowing. This generalisation is done by Jouannaud *et al.* (1982), where this lemma is used for the construction of unification algorithms.

We can then extend Theorem 5.1 (consequence-finding completeness) by replacing narrowing by P-narrowing, blocked resolution by blocked P-resolution and subsumption by P-subsumption. For this extension, we have to use Plotkin's completeness results about P-resolution (1972).

All other results of Sections 5.1 and 5.2 are carried over without difficulty. In particular, the confluence criterion of Jouannaud can be applied to prove the confluence of the system $RS1$ used in proof of Lemma G.

Note that in this framework, the set of non-oriented equations is the union of two systems:

—The equational system P.
—The set of associativity/commutativity equations for the Boolean connectors v and &.

5.4. BIRKHOFF'S THEOREM

In this section, we extend the results of Section 4 to first order predicate calculus in an equational theory.

Let T be an equational theory, and $S = \{E1, \ldots, En\}$ a set of quantifier-free formulas which do not contain any equality predicates. $E1, \ldots, En$ are not necessarily in clausal form.

We suppose that we can build from T a canonical term rewriting system (or a canonical and coherent equational term rewriting system as in Section 5.3).

We associate with $T \cup S$ an equational system which is the union of the three following systems:

—The system BOOL defined in Section 2.2 (regarded as a set of equations).
—The equational system T.
—The set of equations $\{Ei = \text{TRUE}\}$ with Ei in S.

Note that the symbol $=$ corresponds in this system either to the Boolean connector $\langle = \rangle$ or to the equality predicate.

This equational system defines an equality relation on the set of quantifier-free formulas without equality predicates. We denote this relation $=_{TS}$.

THEOREM 5.4. *Let F and G be two quantifier-free formulas without equality predicates. The formula $F \langle = \rangle G$ is a valid consequence of $T \cup S$ in predicate calculus with equality iff $F =_{TS} G$.*

PROOF. The proof follows closely the proof of Theorem 4.1 and is left to the reader. Although this proof is valid only if we can build from T a canonical rewriting system, we conjecture that this theorem is true for any equational theory T.

Note the difference between Theorem 4.1 and Theorem 5.4: in Theorem 5.4, we only deal with formulas which do not contain the equality predicate. In exchange for this restriction, we do not use explicitly the axiomatisation of equality (reflexivity, symmetry, transitivity, substitution). In fact, this axiomatisation is used implicitly when we use an equation $M = N$ in T for replacing in a formula an instance of M by the corresponding instance of N (if we use Theorem 4.1, we can only replace an instance of $(M = N)$ by TRUE).

REMARK. If $T \cup S$ is satisfiable, Theorem 5.4 can be extended to the case where F and G are two non-Boolean terms M and N, the Boolean connector $\langle = \rangle$ being replaced by the equality predicate. We do not give the detailed proof here (the main step consists in proving that $T \cup S \models M = N$ iff $T \models M = N$; then the result follows from the ordinary Birkhoff theorem applied to the equational theory T).

Therefore, Theorem 5.4 is true for any expressions (Boolean formulas or non-Boolean terms) without equality predicate.

5.5. GENERAL CONFLUENCE IN FIRST ORDER PREDICATE CALCULUS

From Theorem 5.4, we obtain a method for building rewriting systems confluent on formulas without equality predicate, based on the Knuth & Bendix completion algorithm and the Hsiang system for propositional calculus. Theorems 4.2 (for general confluence) and 4.3 (for refutational proof) are easily extended in this framework.

Note that if the canonical rewriting system for the equational theory T is not provided at the beginning, we can run the completion algorithm simultaneously on T and S.

EXAMPLE. Let us consider the following specification:

Equational theory T:

$$X+0 = X, \qquad X+Y = Y+X, \qquad X+(Y+Z) = (X+Y)+Z.$$

Set of formulas S:

$$\neg(0 > X), \qquad X+1 > 0, \qquad X+1 > Y+1 \langle = \rangle X > Y.$$

Applying the Knuth & Bendix completion algorithm, we obtain the following system, complete w.r.t. the associativity/commutativity of $+$, !, &:

(Hsiang's system $+$)

$$\begin{aligned} X+0 &\to X \\ 0 > X &\to \text{FALSE} \\ 1 > 1 &\to \text{FALSE} \\ 1 > 0 &\to \text{TRUE} \\ X+1 > Y+1 &\to X > Y \\ X+1 > 0 &\to \text{TRUE} \\ 1 > X+1 &\to \text{FALSE} \\ X+1 > 1 &\to X > 0 \end{aligned}$$

If we use the (resolution + narrowing) algorithm of Theorem 5.3, we must split the equivalence into two clauses. We obtain an infinite rewriting system:

(system BOOL+)

$$\begin{aligned} X+0 &\to X \\ \neg(0 > X) &\to \text{TRUE} \\ X+1 > 0 &\to \text{TRUE} \\ \neg(X+1 > Y+1) \vee (X > Y) &\to \text{TRUE} \\ \neg(X > Y) \vee (X+1 > Y+1) &\to \text{TRUE} \\ \neg(X+1+1 > Y+1+1) \vee (X > Y) &\to \text{TRUE} \\ \neg(X+1+1+1 > Y+1+1+1) \vee (X > Y) &\to \text{TRUE} \\ \ldots & \end{aligned}$$

6. Conclusion

We think that the results presented in this paper can contribute to a better understanding of the relationships between equational methods and first order predicate calculus. Of course, from a practical point of view, classical refutational techniques (based on resolution or on the Knuth–Bendix algorithm) are obviously more efficient than direct proofs such as the ones suggested in this paper, because the research of the proof can then be directed by the theorem to be proved. Another advantage of refutational techniques is that many methods are available to decrease the number of formulas generated, such as the numerous restrictions of resolution (Chang & Lee, 1973), or, if we use the Knuth–Bendix algorithm, the restrictions proposed by Hsiang in the computation of critical pairs (Hsiang & Dershowitz, 1983).

These restrictions can be applied because in refutational methods, only a contradiction (the empty clause, or the equation TRUE = FALSE) is being sought. When using the methods described in this paper, all valid consequences of the theory are generated, or more precisely, a rewriting system confluent on all valid consequences is constructed. Actually, it seems very rare that a first order theory could be compiled into a finite confluent rewrite system, except for rather trivial theories such as those given in Section 4.3.2. Therefore the problem is much more difficult than the word problem in equational theories, for which many canonical rewriting systems are known (for group theory, ring theory, etc.).

Note also that, if the theory or the theorem to be proved in this theory involves existential quantifiers, only refutational methods can be used. That is due to the fact that a closed first order formula F cannot be assimilated to the equation $F = \text{TRUE}$ if some variables of F are not universally quantified.

An open problem is the extension of the results of Section 5 to the case where equality occurs positively in some non-unit clauses defining the theory (as in Lankford's paper (1975), all these results are easily extended to the case where equality occurs only negatively in non-unit clauses, provided that we add the equality reflexivity clause $x = x$, considered as the equation $(x = x) = \text{TRUE}$, to the theory).

Such theories can be represented in a natural way by conditional equations and rewrite rules [see Remy (1982)]. A Birkhoff-like theorem for such theories would provide a strong theoretical basis for studies in this area, in the same way as the Birkhoff theorem for equational theories is the basis for the study of rewriting systems. Such a theorem can be conceived as a completeness result for a certain combination of cases analysis and equational deduction, such as the combination used in Remy (1982).

References

Birkhoff, G. (1935). On the structure of abstract algebras. *Proc. Cambridge Phil. Soc.* **31,** 433–454.

Bucken, H. (1979). Reduction systems and small cancellation theory. *Proc. Fourth Workshop on Automated Deduction,* 53–59.

Chang, C. L. & Lee, R. C. (1973). *Symbolic Logic and Mechanical Theorem Proving.* New York: Academic Press.

Fages, F. (1983). *Formes Canoniques dans les Algèbres Booléennes, et Application à la Demonstration Automatique en Logique du Premier Ordre.* Thèse de 3me cycle, Université Pierre et Marie Curie.

Hsiang, J., Dershowitz, N. (1983). Rewrite methods for clausal and non-clausal theorem proving. *Proc. ICALP 83, Spain.*

Huet, G. (1981). A complete proof of correctness of the Knuth–Bendix completion algorithm. *J. Comp. Syst. Sci.* **23, 1,** 11–21.

Huet, G., Oppen, D. C. (1980). Equations and rewrite rules: a survey. In: *Formal Languages: Perspectives and Open Problems.* New York: Academic Press.

Hullot, J. M. (1980). Canonical form and unification. Proc. Fifth Conference on Automated Deduction, Les Arcs. In: (Bibel, W., Kowalski, R., eds) *Lecture Notes in Computer Sciences 87.* Heidelberg: Springer-Verlag.

Jouannaud, J. P., Kirchner, C., Kirchner, H. (1982). Incremental unification in equational theories. *Proc. of the 21st Allerton Conference.*

Jouannaud, J. P. (1983). Confluent and coherent sets of reduction with equations. Application to proofs in data types. *Proc. 8th Colloquium on Trees in Algebra and Programming.*

Knuth, D., Bendix, P. (1970). Simple word problems in universal algebra. In: (Leech, J., ed.) Computational problems in abstract algebra, pp. 263–297. Oxford: Pergamon Press.

Lankford, D. S. (1975). *Canonical Inference.* Report ATP-32, Department of Mathematics and Computer Science, University of Texas at Austin.

Lankford, D., Musser, D. R. (1978). *On Semi-deciding First Order Validity and Invalidity.* USC–ISI Report.

Lee, R. C. (1967). *A Completeness Theorem and a Computer Program for Finding Theorems Derivable for Given Axioms.* Ph.D. diss. in engineering, Univ. of California, Berkeley, Calif.

Paul, E. (1984). Proof by induction in equational theories with relations between constructors. In: (Courcelle, B., ed.) Proc. 9th Colloquium on Trees in Algebra and Programming, Bordeaux (March 1984). Cambridge: Cambridge Univ. Press.

Peterson, G., Stickel, M. (1981). Complete sets of reductions for some equational theories. *J. Assoc. Comp. Mach.* **28,** 233–264.

Plotkin, G. (1972). Building-in equational theories. In: (Meltzer, N., Michie, D., eds) *Machine Intelligence No. 7,* pp. 73–90. Edinburgh: Edinburgh Univ. Press.

Remy, J. L. (1982). *Etude des Systèmes de Réecriture Conditionnels et Application aux Types Abstraits Algèbriques.* Thèse docteur es sciences, Centre de Recherche en Informatique de Nancy, July 1982.

Robinson, J. A. (1965). A machine-oriented logic based on the resolution principle. *J. Assoc. Comp. Mach.* **12,** 23–41.

Slagle, J. R. (1974). Automatic theorem proving for theories with simplifiers, commutativity and associativity. *J. Assoc. Comp. Mach.* **21**, 622–642.
Tarski, A. (1946). A remark on functionally free algebras. *Ann. Math.* **47**, 163–165.

Appendix

It is proved in this appendix that the computation of binary factors and binary resolvents can be considered as a computation of critical pairs restricted to certain critical pairs. To take into account the associativity/commutativity of v, we consider for each rule its extension rule, as defined by Peterson & Stickel (1981).

1. BINARY FACTOR

C being a clause and $F(C)$ a binary factor of C, the pair $\langle F(C), \text{TRUE}\rangle$ is obviously a critical pair between the rule $C \to \text{TRUE}$ and the rule $X \vee X \to X$ (or its extension $Y \vee X \vee X \to Y \vee X$) of BOOL.

2. BINARY RESOLVENT

Let $C = C1 \vee L$ and $D = D1 \vee \neg P$ be two clauses, L and P being two positive unifiable literals. Let σ be the mgu of L and P. A binary resolvent of C and D is $R(C, D) = C1\sigma \vee D1\sigma$. The rewrite rules associated with C and D are:

$$C1 \vee L \to \text{TRUE}$$
$$D1 \vee \neg P \to \text{TRUE}$$

We suppose that $C1$ and $D1$ are not the empty clause (i.e. C and D are not unit clauses). The proof is easily extended if $C1$ and/or $D1$ are the empty clause.

The pair $\langle R(C, D), \text{TRUE}\rangle$ can be obtained by a computation of critical pairs as follows:

The rules of BOOL:

$$X \vee (Y \,\&\, Z) \to (X \vee Y) \,\&\, (X \vee Z)$$
$$X \,\&\, \neg X \to \text{FALSE}$$

can be superposed, generating the rule:

$$(X \vee Y) \,\&\, (X \vee \neg Y) \to X \tag{1}$$

Rule (1) can be superposed with the rule $C1 \vee L \to \text{TRUE}$, generating the rule:

$$C1 \vee \neg L \to C1. \tag{2}$$

The rule (2) can be considered as a kind of "extension" of the rule $C1 \vee L \to \text{TRUE}$. Such additional rules are also used by Hsiang & Dershowitz (1983) and Fages (1983) for computing critical pairs.

The two rules:

$X \vee C1 \vee \neg L \to X \vee C1$ (extension of rule (2))
$Y \vee D1 \vee \neg P \to \text{TRUE}$ (extension of rule $D1 \vee \neg P \to \text{TRUE}$)

can be superposed since L and P are unifiable. The rule generated is: $C1\sigma \vee D1\sigma \to \text{TRUE}$, i.e. $R(C, D) \to \text{TRUE}$.

Note that we retain only the rule $R(C, D) \to \text{TRUE}$ to build the system $R\infty$ confluent on valid formulas. It is not necessary to retain the intermediate rules (1) and (2).